# Cost Reduction Bounds of Proactive Management Based on Request Prediction

Ruben Milocco*, Pascale Minet†, Éric Renault‡ and Selma Boumerdassi§

* GCAyS, UNComahue, Buenos Aires 1400, 8300 Neuquén, Argentina. Email:milocco@uncoma.edu.ar
† Inria, 2 rue Simone Iff, CS 42112, 75589 Paris Cedex 12, France. Email: pascale.minet@inria.fr
‡ SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, 91011 Évry, France. Email: eric.renault@telecom-sudparis.eu
§ CNAM/CEDRIC, 292 rue Saint Martin, 75003 Paris, France. Email: selma.boumerdassi@cnam.fr

**Abstract**

Data Centers (DCs) need to manage their servers periodically to meet user demand efficiently. Since the cost of the energy employed to serve the user demand is lower when DC settings (e.g. number of active servers) are done *a priori* (proactively), there is a great interest in studying different proactive strategies based on predictions of requests. The amount of savings in energy cost that can be achieved depends not only on the selected proactive strategy but also on the statistics of the demand and the predictors used. Despite its importance, due to the complexity of the problem it is difficult to find studies that quantify the savings that can be obtained. The main contribution of this paper is to propose a generic methodology to quantify the possible cost reduction using proactive management based on predictions. Thus, using this method together with past data it is possible to quantify the efficiency of different predictors as well as optimize proactive strategies. In this paper, the cost reduction is evaluated using both ARMA (Auto Regressive Moving Average) and LV (Last Value) predictors. We then apply this methodology to the Google dataset collected over a period of 29 days to evaluate the benefit that can be obtained with those two predictors in the considered DC.

**Index Terms**

data centers management, proactive management, machine learning, prediction, energy cost, ARMAX

## I. Introduction

The problem of managing Data Centers (DC) and clouds optimally, in the sense that the user demand can be met with a minimal energy cost, remains a major issue [1]. Within this paradigm, the use of predictive models could be useful to improve the management. Previous studies have proposed DC management models. In the scope of this paper, we evaluate the energy cost reduction brought by predicting job arrivals and their resource requests in terms of processing capacity and memory, considering that having knowledge of the number of requests to come can greatly help job placement and scheduling, thus DC node management, and then reduce the global energy consumption of the DC.

Diverse approaches to obtain predictive models of DCs have been studied recently, as we will see in Section II. Among the most popular methods with the comparatively lowest prediction errors are the predictive ARMAX (ARMA with eXogenous signal) family models. **In this paper, we study the bounds of the improvement in terms of energy cost that can be obtained using proactive strategies for DC management based on predictive models.** The savings that can be obtained are numerically evaluated on a real DC dataset [2], [3], made publicly available by Google. These traces were collected in an operational DC over 29 days. Having traces of a real DC is an excellent opportunity for researchers to improve DC management, obtain models for testing placement and scheduling algorithms before their deployment in real DCs.

This paper is organized as follows. In Section II, the state of the art in request prediction is described. In Section III, the energy cost is defined as the sum of two components: a proactive one based on the request prediction, and a reactive one which is paid only in case of request under-estimation. With this energy cost model, we compare the energy cost reduction brought by an ARMAX predictor and this given by the Last Value (LV) predictor, which predicts that the next value will be the same as the current one. Based on the Last Value predictor, which corresponds to pure reactive action, the method for obtaining the bound of energy saving attainable is presented. In Section IV, the ARMAX family of predictive models, which is among the most accepted models for request prediction, is analyzed. The conditions of linearity, steady state and residence time, which are all needed to apply ARMAX models, are expressed. In section V, the parameters of these ARMA models are tuned recursively, based on the Recursive Prediction Error Method (RPEM), to be used online using the Google dataset. The savings that can be obtained are also quantified. Finally, we conclude the paper in Section VI.

The bounds computed with the energy cost model proposed in Section III can be applied to any operational DC in steady state. The values of these bounds depend on the DC considered. With these bounds, one can determine when the LV predictor is sufficient and when a more sophisticated predictor should be used to reduce the energy cost.

## II. State of the art

There are several studies on request prediction strategies to improve DC management using machine learning techniques such as linear predictive models, probabilistic approaches, neural networks, among others. In [4], Liu et al. using the dataset from the Google data center conclude that the Moving Average (MA), AutoRegressive (AR), and the Weighted Moving Average (WMA) prediction models give the smallest prediction errors with the lowest complexity compared to neural networks. We recall that AR and MA belong to the family of AutoRegressive Moving Average with Exogenous signal (ARMAX) models. In [5], Prevost et al. evaluate the predictions provided by neural networks and AR models on the URL resource requests of a www server at NASA and EPA. They conclude that AR is far superior to neural networks. In [6], Yoon et al. define a prediction model that estimates parameters of a Poisson distribution with Local Linear Regression (LLR) by using a cyclic window approach. Using this method to predict the number of job arrivals in the next 30 minutes, the authors evaluate the prediction accuracy of their method by the Mean Absolute Percentage Error (MAPE), and they obtain a value of 0.38, which is similar to that obtained using ARMA models. In [7], several variants of ARMA models (such as integrated, fractional, seasonal) are analyzed and compared with an exponential smoothing model and a spectral estimation method based on singular spectrum analysis. The methods were used to compare multiple time series models for predicting CPU, RAM, and network consumption over a one-hour time interval. They conclude that ARIMA models outperform the others. In [8], ARMA models used to predict the popularity of video content on YouTube, are shown to achieve better performances than other prediction models.

A data set collected in a Google data center over a period of 29 days and published in [2], [3], has been analyzed by many researchers [9]–[14] for both classification and prediction. In [9], K-Means is used to classify the jobs into three categories according to their duration: short, medium and long. Short jobs are shown to be the most frequent and use fewer resources. Medium jobs are frequent and have strong requirements in terms of memory. Long jobs, the least frequent, have strong requirements in terms of CPU. Using K-Means and density-based clustering, the authors of [12] also classify the tasks into two categories according to their resource requests: CPU-intensive and memory-intensive. Their approach consists in associating a virtual machine with each task, and placing two virtual machines of different categories on the same physical machine. However, there is no performance evaluation to validate this approach. In [13], the authors use a Bayesian model to predict the average host load. The authors show that the Bayesian method improves the ARMA prediction error by 5% at 6 hours, and 8% at 12 hours. In [14], the features of jobs submitted in the Google data center are studied.

## III. Cost and prediction

Each request arrives in the DC at a random time, and the goal is to predict future requests based on the history of the past values. The prediction of such individual events is extremely complicated as time intervals between them are statistically independent of each other. One possible strategy to solve this problem is to split the time into fixed length intervals of size $T$, and accumulate the number of requests during each interval, as shown in Fig. 1. $T$ is called the accumulation interval. To simplify the notation, we call step $k$, with $k \geq 1$, the $k^{th}$ accumulation interval corresponding to times $t$ with $(k-1)T \leq t < kT$. Let $R(t)$ denote the requests arriving at time $t$. We define $y(k)$ the number of requests arriving during step $k$ as

$$y(k) = \sum_{(k-1)T \leq t < kT} |R(t)|, \tag{1}$$

where $|R(t)|$ denotes the cardinality of set $R(t)$.

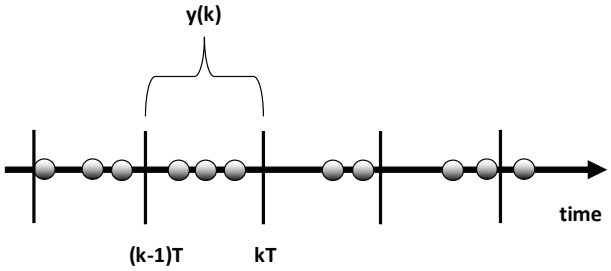


Fig. 1. Time discretization.

We also adopt the energy model from [15] and [16], where the power $P(k)$ consumed by a server during step $k$ is given by the following equation:

$$P(k) = P_0 + \eta \times Load(k) \tag{2}$$

where $P_0$ denotes the power consumption of a server at null load and $\eta$ is the power variation coefficient as a function of the load during step $k$. The load is proportional to the number of arriving requests $y(k)$. In addition, the DC operation guarantees

the average service time of requests is time-invariant. Then in this work, energy is considered proportional to the power and also to the arriving requests. To maintain these stationary characteristics the DC is reconfigured every time period $T$ either by adding or removing machines, by performing migrations, etc., in order to optimally serve the requests.

Let us consider the cost of energy at step $k$ used to serve the $y(k)$ incoming requests as the sum of two components. The first one is the cost of energy due to the proactive action which consists in serving $\hat{y}(k)$, the predicted incoming requests at step $k$, given information until step $k-1$. In this case, the DC is configured a priori, i.e. at time $(k-1)T$, by expecting $\hat{y}(k)$ requests in the next time interval $T$. The second, is due to the reactive operation which consists in serving $\max(0, y(k)-\hat{y}(k))$ requests a posteriori, i.e. at time $kT$, to correct the prediction error made for step $k$, in case of under-estimation. Only in this case, reactive actions are needed. In case of over-estimation, the number of available servers is higher than needed, the users will be served immediately; however, some energy is wasted, as reflected by the proactive cost. Whatever the prediction, the proactive cost is assumed to be smaller than the reactive one.

For example, assume that at step $k$ there are $y(k) = n$ incoming requests and that the predicted value is $\hat{y}(k) = n - m$, with $m > 0$. Then, the total cost at step $k$ is equal to the proactive cost of serving $n - m$ requests plus the correction after step $k$ paying a reactive cost of serving $m$ requests. The cost of the reactive action is higher than the cost of the proactive action because proactive actions allow optimizing resources much better than corrective ones. In addition, in case of under-estimation, some requests have to wait until their resources are allocated, whereas in the proactive case, their resources are immediately available. Then, the total cost can be represented by the following cost function:

$$J(k, \alpha) = \alpha\hat{y}(k) + (1-\alpha)\max(0, y(k) - \hat{y}(k)) \tag{3}$$

where $J(k, \alpha)$ is the cost of serving the incoming requests $y(k)$ at step $k$ and $\alpha \in (0, 1/2)$ weighs the difference between the cost of the proactive and reactive actions.

The cost can be computed for a given horizon of $N \geq 1$ past accumulation intervals, where $k = 1, 2, \ldots, N$. Using a simple notation to express the energy cost per request, also called the relative cost $J_x(\alpha)$, for a prediction model $x$, we get:

$$J_x(\alpha) = \alpha\|\hat{Y}_x\|_1 + (1-\alpha)\|\max(0, E_x)\|_1 \tag{4}$$

where $\|\hat{Y}_x\|_1$ and $\|\max(0, E_x)\|_1$ are the $L_1$ norms of the sequences $\hat{y}_x(k)$ and $\max(0, e_x(k))$ defined as: $\|Y_x\|_1 = \sum_{k=1}^{N} |\hat{y}_x(k)|$, $\|\max(0, E_x)\|_1 = \sum_{k=1}^{N} \max(0, e_x(k))$, where $e_x(k) = y_x(k) - \hat{y}_x(k)$ is the prediction error for a given a prediction model $x$. We consider two prediction models: the ARMA (ARMAX) predictor denoted by the index $x = p$ and the Last Value predictor (LV) denoted by the index $x = r$, which predicts that the next value will be equal to the current value, $\hat{y}_r(k) = y(k-1)$. Similarly, the errors $e_p(k) = y(k) - \hat{y}_p(k)$ and $e_r(k) = y(k) - \hat{y}_r(k) = y(k) - y(k-1)$ stand for the ARMA and LV prediction errors, respectively.

It is important to note that in the case of the LV prediction, there is no proactive action itself since there is no decision making prior to the step $k$, because the prediction is just the same value as at the step before. In other words, if the prediction is equal to the real value, there is no need of proactive actions. Thus, provided that there are predictors such that $J_r(\alpha) \geq J_p(\alpha)$, the difference between the cost using the last value prediction and the cost using the ARMA predictor is in fact a bound of cost saving using prediction. The relative cost improvement can be written as:

$$\Delta(\alpha) = \frac{J_r(\alpha) - J_p(\alpha)}{\|Y\|_1} \tag{5}$$

where the $L_1$ norm of the number of incoming requests during the last $N$ time intervals, $\|Y\|_1 = \sum_{k=1}^{N} |y(k)|$, is used in order to have the cost reduction per request. We assume that the Data Center is operated in steady state, which means that the average input flow of requests to the DC is equal to the average output flow. This condition depends on the length of the accumulation interval $T$ and also on the average service time. It is shown in Section V that this condition is met by an efficient management of data centers resources. The flow of requests is slowly time-varying for large values of the accumulation interval $T$, but highly fluctuating when $T$ is reduced. In this way, if corrective actions are done at small intervals, the cost may be considerably reduced with respect to the case of corrections done at greater intervals $T$.

Finally, it should be pointed out that there are several optimal solutions to minimize the $L_1$-norm problem of cost $J_p(\alpha)$. In this paper, we obtain the pseudo-optimal solution which consists in minimizing the $L_2$ norm of the error, which is unique using the standard minimum least squares method. We evaluate the costs defined by equation (4) and the relative cost improvement defined by equation (5) using experimental data, but first we present the prediction methods in the next section.

## IV. How to predict

Many predictors have been developed so far to forecast the state of a system. In the scope of our work, we focus on two of them that demonstrated their efficiency: ARMA and ARMAX. Later in the article, we analyze how much one can reduce the prediction error with respect to the classical ARMA model when several sources of information are available. Then, this section briefly presents the recursive algorithm that solves the least squares problem. The new contribution is the identification

of the model parameters that produce minimum norm 1 of the error, in order to obtain a predictor that minimize the energy cost. Readers who are familiar with predictors could probably skip this section.

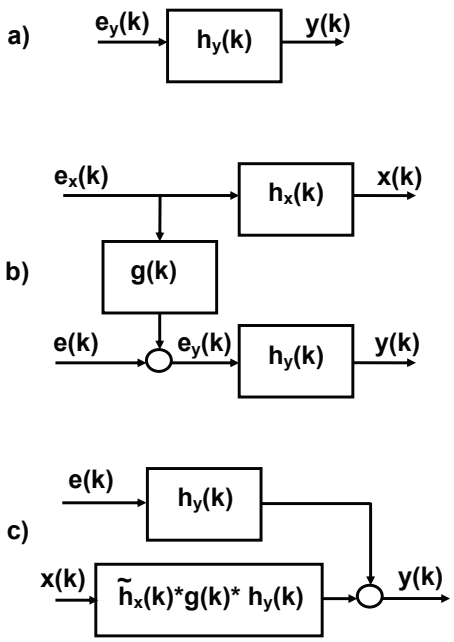


Fig. 2. a) Single input-single output ARMA model, b) two input-two output ARMA model, c) two input-single output ARMA model.

### *A. The ARMA model*

Fig. 2 depicts three variants of ARMA models that will be used in this paper. The first ARMA variant, depicted in Fig. 2.a, corresponds to a single input and a single output, where the flow is modeled as the convolution between the impulse response $h_y(k)$ and an $i.i.d$ sequence $e_y(k)$ with zero mean and unit variance as follows:

$$y(k) = \sum_{i=0}^{n} h_y(i) e_y(k-i) = h_y(k) * e_y(k), \tag{6}$$

where $*$ means convolution and $h_y(k)$ is the impulse response of an invertible-stable-causal linear system, also called a minimum phase system. A scheme is given in Fig 2a. The minimum-phase property guarantees that there exists another causal and stable sequence, $\tilde{h}(k)$, called the inverse of $h(k)$, such that $\tilde{h}_y(k) * h_y(k) = \delta(k)$, where $\delta(k)$ is the unit impulse sequence [17]. The sequence $\tilde{h}_y(k)$ is unique. This kind of system can be represented by an AutoRegressive Moving Average (ARMA) model having the following structure, where $na$ and $nc$ are the orders of past values of $y(k)$ and $e(k)$, respectively, whereas $a_i$ and $c_j$, for $i = \{1, \ldots, na\}$ and $j = \{1, \ldots, nc\}$ are coefficients computed at each step $k$:

$$\begin{aligned} y(k) =& a_1(k)y(k-1) + \cdots + a_{na}(k)y(k-na) \\ &+ e_y(k) + c_1(k)e_y(k-1) + \cdots + c_{nc}(k)e_y(k-nc). \end{aligned} \tag{7}$$

Now, considering a prediction horizon equal to one step, the sequence $h_y(k)$ in equation (6) can be written as the sum of the anticausal, $h_y(0)$, and the causal, $h_y^+(k) = \{h_y(1), h_y(2), \ldots\}$ components such that the number of requests arrived during step $k+1$ is written as:

$$\begin{aligned} y(k+1) &= h_y(k) * e_y(k+1) \\ &= h_y(0)e_y(k+1) + h_y^+(k) * e_y(k). \end{aligned} \tag{8}$$

The second term on the right is the causal part in which, replacing $e_y(k)$ by $\tilde{h}_y(k) * y(k)$, we obtain the minimum variance one-step ahead predictor $y(k+1|k)$, given data until step $k$. The first term, the anticausal part, is the prediction error having variance equal to $h_y^2(0)$. Hence, we finally get:

$$y(k+1) = h_y(0)e_y(k+1) + \hat{y}(k+1|k) \tag{9}$$

The two models depicted in Fig. 2.b and Fig. 2.c are explained in the next subsection, which deals with the ARMAX model.

### B. ARMAX model

Due to the possible relationship between different requests, is-it possible to improve the predictions of some requests knowing the values of other requests? The problem is posed in the framework of the Wiener-Granger causality concept, [19]: if a stationary stochastic sequence $x(k)$ contains information in past time that improves the predictions of a sequence $y(k)$, then $x(k)$ is said to Granger-cause $y(k)$. Such causality can be known by analyzing the minimum phase property of the system that relates the input $x(k)$ to the output $y(k)$. Basically, if the system between $x(k)$ and $y(k)$ is causal and stable and the inverse of the system is also causal and stable (minimum phase systems) predictions cannot be improved, since the input $x(k)$ can be known from measurements $y(k)$ and the system inverse. On the other hand, if the system is not inverse-causal and stable (non-minimum phase systems), the measurement $x(k)$ along with the dynamics of the system can improve the prediction of $y(k)$.

For a formal setup we refer to Fig. 2.b. Let us assume that $x(k)$ and $e_y(k)$ can be expressed as:

$$x(k) = h_x(k) * e_x(k) \tag{10}$$
$$e_y(k) = e(k) + g(k) * e_x(k), \tag{11}$$

where $h_x(k)$ is the impulse response of a minimum phase system; $e(k)$ and $e_x(k)$ are i.i.d zero mean sequences independent of each other. The sequence $e_x(k)$ has unit variance and $g(k)$ is an all-pass filter with gain $\gamma$ such that $\sum_k g^2(k) = \gamma^2 \leq 1$, thereby $e(k)$ has variance $1-\gamma^2$. Using our notation for convolution in equation (6) and replacing equation (11) in equation (8), the request at step $k+1$ is written as:

$$y(k+1) = h_y(k) * (e(k+1) + g(k) * e_x(k+1)), \tag{12}$$

Note that similarly to equation (8), equation (12) is a sum of a causal part and an anti-causal part. By replacing $e_x(k)$ by $\tilde{h}_x(k) * x(k)$, where $\tilde{h}_x(k)$ is the inverse of $h_x(k)$, in the causal part we obtain the minimum variance one-step ahead predictor $\hat{y}(k+1)$, given data until step $k$ using requests $x(k)$ and $y(k)$. The anti-causal part is the prediction error and is given by:

$$h_y(0)(e(k+1) + \sum_{i=-\infty}^{0} g(i) e_x(k+1-i)). \tag{13}$$

Note that the sum in the convolution includes the anti-causal part of $g(k)$. Taking into account that $e(k)$ is independent of $e_x(k)$, its variance is:

$$h_y^2(0)(1 - \gamma^2 + \sum_{i=-\infty}^{0} g^2(i)). \tag{14}$$

where the last term is always less than or equal to $\gamma^2$. Thus, the ARMAX predictor outperforms the ARMA predictor, except when $g(k) = \gamma\delta(k)$ (minimum phase all-pass) or when $\gamma^2 = 0$ meaning that $x(k)$ is independent of $y(k)$.

The scheme of Fig. 2.b is equivalent to the AutoRegressive Moving Average with eXogenous signal (ARMAX) model as depicted in Fig. 2.c. The ARMAX model has the following structure, where $na$, $nb$ and $nc$ are the orders of past values of $y(k)$, $x(k)$, and $e(k)$, respectively, whereas $a_i$, $b_j$ and $c_h$, for $i = \{1, \ldots, na\}$, $j = \{1, \ldots, nb\}$ and $h = \{1, \ldots, nc\}$ are coefficients computed at each step $k$,

$$\begin{aligned} y(k) =& a_1(k)y(k-1) + \cdots + a_n(k)y(k-na) \\ &+ b_1(k)x(k-1-nk) + \cdots + b_n(k)x(k-nk-nb) \\ &+ e(k) + c_1(k)e(k-1) + \cdots + c_n(k)e(k-nc), \end{aligned} \tag{15}$$

where $nk$ is the delay of the exogenous signal $x(k)$ and the input signal $e(k)$ is assumed to be i.i.d. stochastic processes with normal distributions $e_k \backsim \mathcal{N}(0, \sigma_e^2(0))$. Note that when $x(k) = 0$ the structure turns into an ARMA model.

Then, the prediction problem can be formulated as an identification of ARMA or ARMAX models to obtain the one-step ahead prediction $\hat{y}(k+1)$, based on the knowledge of previous values for steps $k, k-1, k-2, \cdots$, where their parameters are identified by the Recursive Prediction Error Method (RPEM) [18], detailed in the next Section.

### C. RPEM

Let us assume the following quadratic cost over an horizon of $k$ past samples:

$$V_k = \frac{1}{2} \sum_{i=1}^{k} \lambda^{k-i} \epsilon_i^2 \tag{16}$$

where $\epsilon_i$ is the prediction error defined in equation (17) and $\lambda$ is a scalar in the interval $(0, 1]$ called the *forgetting factor*, which performs an exponential windowing over the previous prediction errors. The width of the exponential window depends

on the value of $\lambda$. If $\lambda < 1$, previous prediction errors contribute only marginally to the criterion function. The window's width is reduced as $\lambda$ decreases. If $\lambda = 1$, all past data are equally weighted. Thus, the value of $\lambda$ determines the memory of the past data, which is a suitable parameter to take into account time-variant mobility dynamics. The prediction error is given by:

$$\epsilon_k = y(k) - \hat{y}(k) \tag{17}$$

where $\hat{y}(k) = f(k)\theta(k)$ is the ARMAX model written as a linear regression prediction problem with $f(k) = [y(k-1), \cdots, y(k-na), x(k-1-nk), \cdots, x(k-nk-nb), e(k), e(k-1), \cdots, e(k-nc)]$, and parameters vector $\theta_k = [a_1(k), \ldots, a_n(k), b_1(k), \ldots, b_n(k), c_1(k), \ldots, c_n(k)]^T$ that minimizes $V_k$. The parameters vector $\theta_k$ is unknown and it is estimated by using the well-known Prediction Error Method (RPEM). The algorithm and its properties are given by the following theorem:

$Theorem$: Consider the cost function $V(k)$ to be minimized with respect to the parameter vector $\theta(k-1)$ by the following Gauss-Newton recursion:

$$\begin{aligned}
\hat{y}(k) &= f(k)\hat{\theta}(k-1) \\
\epsilon_k &= y(k) - y(k|k-1) \\
\varphi(k) &= -\epsilon'(k) \\
M(k) &= M(k-1) - \frac{M(k-1)\varphi(k)\varphi^T(k)M(k-1)}{1+\varphi^T(k)M(k-1)\varphi(k)} \\
\theta(k) &= \theta(k-1) + M(k)\varphi(k)\epsilon(k)
\end{aligned}$$

where $k$ is the iteration step, $M_k$ is a square matrix of dimension $(2n)$; and $\epsilon_k^{'}$ is a column vectors derivative of $\epsilon_k$ with respect to the parameters in $\theta_k$. Then, $\theta_k$ converges as $k \to \infty$ with probability 1 to one element of the set of minimizers

$$\left\{\theta | \sigma_\epsilon^{'2} = 0\right\}; \tag{18}$$

where $\sigma_\epsilon^{'2}$ is the derivative of the prediction error variance with respect to $\theta$.

Proof : See [18].

## V. Prediction on the Google dataset

### A. Dataset properties

The dataset used in this study [3] comes from traces collected in a Google data center over a period of 29 days. It represents about 12,000 servers and 670,000 jobs. Data are organized into 2002 files, representing 179.4 GBytes. Files are grouped into 6 tables dealing with machine events, machine attributes, job events, task events, task constraints and task usage, respectively.

Since more than 92% of jobs have a single task and more than 95.75% of jobs have fewer than 10 tasks, we can assume as a first approximation that all jobs have a single task. That is why in this paper, we focus on the number of incoming jobs.

In addition, since only 1.54% of jobs have a CPU request larger than 10% of the capacity of the most powerful machine, and only 1.74% of jobs have a memory request larger than 10%, we assume as a second approximation that all jobs have similar requests. With these simplifying assumptions, we are able to express the energy cost of a DC as a function of the number of incoming jobs.

From the original data, flow sequences are obtained according to the accumulation time interval $T$ depicted in Fig. 1.

In analyzing job arrivals, we observed a periodic pattern in the incoming number of jobs, as shown in the autocovariance function depicted in Fig. 3. From this figure it is possible to distinguish three periods, $T = 300s$, $T = 3600s$, and $T = 86400s$, corresponding to 5 minutes, 1 hour and 24 hours, respectively. Since these durations are frequently used in human reasoning, it is natural to find them in job submissions.

### B. Prediction models

In order to predict the number of incoming jobs, the ARMAX predictor model is used with several exogenous signals like the number of outgoing jobs, the number of CPU requests and the number of memory requests; but no significant improvements with respect to the ARMA prediction were obtained, except in the case of predictions of output jobs using the number on input jobs as the exogenous signal, as we see later. The optimal order for all of them was found to be $na = 1$ and $nc = 2$ with a forgetting factor of 0.99.

Now we show that the ARMAX model is appropriate for predicting the number of outgoing jobs using the number of input jobs as the exogenous signal. To this end, we first compute the impulse response of the system by calculating the lifetime of each job within the DC and computing the distribution of those times, as depicted in Fig. 4. The number of jobs needs to be large in order to allow the normalized distribution to approach the probability density function which is, in fact, the theoretical

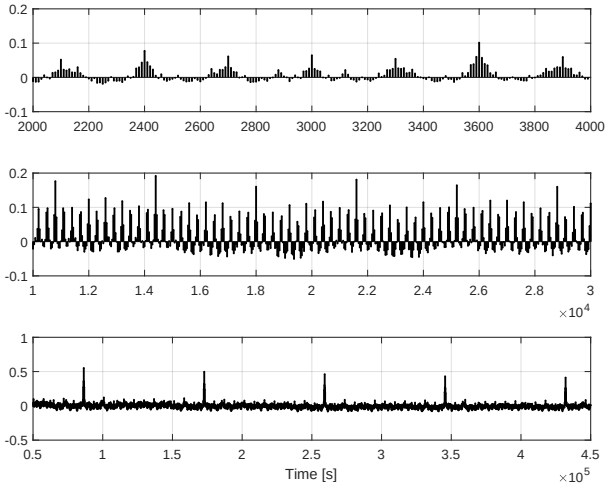

Fig. 3. Covariance function of incoming number of jobs

unit impulse response. The same impulse response is obtained from different sections of the data set which shows that the averaged number of incoming jobs equals the outputs. This steady state condition is the assumption we consider to derive the value of the cost reduction in equation (5).

Using the concept that the impulse response is the distribution of jobs' lifetime, we conclude that the system in steady state is linear with respect to the flow of incoming jobs since it fulfills the superposition theorem. Let us assume that $p$ and $q$ are large numbers of jobs used to compute two histograms, $h_p(k)$ and $h_q(k)$. For example, we consider the number of jobs that enter over five days as $p$ and the following ten days for $q$. The histograms fulfill the superposition theorem which states that the sum of scaled functions is the function of the scaled sum as follows:

$$a\ h_p(k) + b\ h_q(k) = h_{ap+bq}. \tag{19}$$

where $a$ and $b$ are constants. Then, the ARMAX model is appropriate to represent the output number of jobs using the number of incoming jobs as the exogenous signal, a sampling interval of $T = 10s$. First, we use a pure Moving Average (MA) model by setting orders $na = nc = nk = 0$ and $nb = 20$ and $nb = 40$. The identified coefficients correspond exactly to the true impulse response in the number of coefficients considered, as depicted in Fig. 4.i. After that the matching is poor. In order to represent the complete system using an MA model we need more than $nb = 600$ samples, which is impractical. Instead, ARMAX models are used. In this same figure, we depict the responses obtained with $na = 5$. It can be seen that it is a very good approximation.

After having obtained the different approximated systems representing the DC, we observe that all of them are non-invertible causal, and therefore, we expect some improvements in predicting the output flow using the input flow and the identified system. To evaluate such improvements, the performance of the $N$ predictions are evaluated using the mean absolute percentage error (MAPE) computed with the following equation:

$$\text{MAPE} = \frac{1}{N}\sum_{k=1}^{N}\frac{|y(k) - \hat{y}(k)|}{y(k)} \tag{20}$$

In Fig. 4.ii and Fig. 4.iii, the best relative one-step ahead error predictions of input and output flows are depicted, respectively. For the predicted output flow it can be observed that ARMAX improves ARMA prediction for intervals less than $T = 10$ minutes (approximately $10^{2.8}$).

The average lifetime is obtained by computing the complete impulse response, but it has the problem that outliers can shift the average value in such a way that it does not represent the most probable cases. For example, in our case, the average lifetime of all jobs is $612$ seconds, but if we remove these outliers the time changes to $175$ seconds. This value coincides with the transit time of the jobs, which is obtained by identifying the dynamics of the system between input and output numbers of jobs.

### *C. Cost computation*

We now evaluate the costs of reactive and proactive strategies as defined in equation (4) but divided (normalized) by the norm $\|Y\|_1$.

The approach proposed to compute the relative cost consists of the following steps:

- use ARMA to predict the number of incoming jobs;

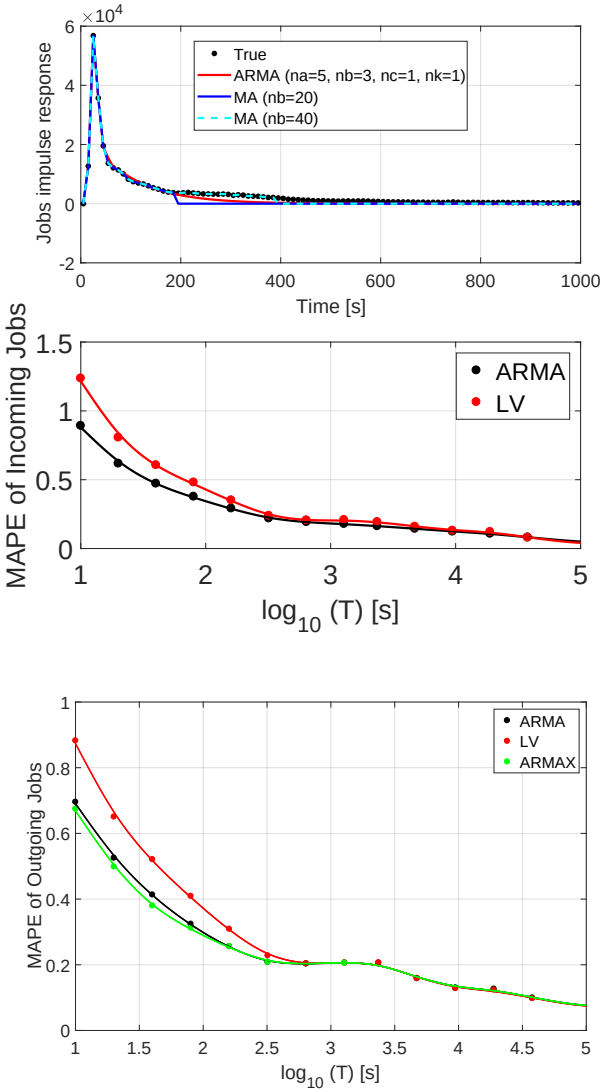


Fig. 4. i) Different estimated responses of outgoing jobs for an impulse of 329960 incoming jobs, ii) MAPE of number of incoming jobs, for ARMA and Last Value predictions, and iii) MAPE of number of ougoing jobs using ARMA, ARMAX with incoming jobs as the exogenous signal, and Last Value models.

- use ARMAX to predict the number of outgoing jobs using the exogeneous signal of the number of incoming jobs;
- compute the load;
- compute the relative cost, that is the cost normalized per request.

This allows to obtain the cost of serving the requests that only depend on the number of incoming jobs.

Figure 5 shows both relative costs, when the time varies from 1 min ($10^{1.8}$ seconds) to 12 hours ($10^{4.6}$ seconds). The value of $\alpha$ in the cost is $0.3$ which means the reactive actions'cost is $7/3$ times the proactive ones. In Fig. 5, we observe that i) the costs decrease when the accumulation interval $T$ increases, and ii) the improvement no longer exists or it is not significant for a time interval larger than 10 minutes (3%) or 20 minutes (2%). This is due to the fact that samples of the sequence are correlated and become more and more correlated as the accumulation interval, $T$, increases. In contrast, as the accumulation interval is reduced, the flow samples are less correlated and the prediction becomes poor. We notice that for $T = 300$ s and $T = 3600$ s, corresponding to $10^{2.48}$ and $10^{3.56}$, the two curves are mixed. This is not surprising since it corresponds to the periodicity of the dataset observed in Fig. 3. For such values of $T$, LV is an excellent predictor.

In Fig. 6, the relative cost improvement $\Delta(\alpha)$, defined in equation (5), is depicted as a function of $\alpha$. For small values of $T$ (i.e. less than 2.5 mn= $10^{2.2}$ $s$), the improvement increases when $\alpha$ (or proactive cost) decreases. In such time intervals the ARMA predictor outperforms LV. In addition, for all the values of $\alpha \in [0.1, 0.4]$, the improvement is maximized for the smallest value of $T$ tested that is 1 mn= $10^{1.8}$ s. For instance, it is equal to 12% when $\alpha = 0.1$. For larger values of $T \geq 2.5$ mn, all the gains are smaller and lesser than 2% which is obtained for $\alpha = 0.4$.

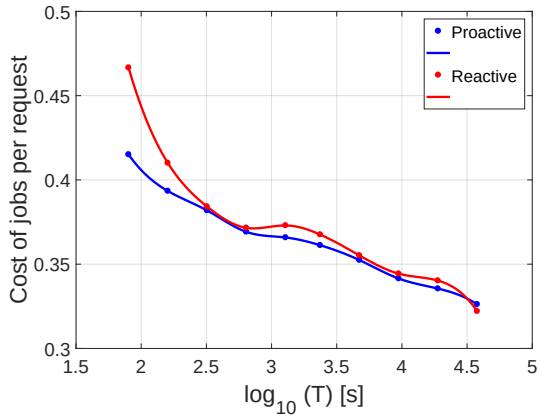


Fig. 5. Costs of the reactive $\frac{J_r(\alpha)}{\|Y\|_1}$ and proactive $\frac{J_p(\alpha)}{\|Y\|_1}$ strategies as a function of interval $T$ in the interval range where proactive strategies improve the reactive ones.

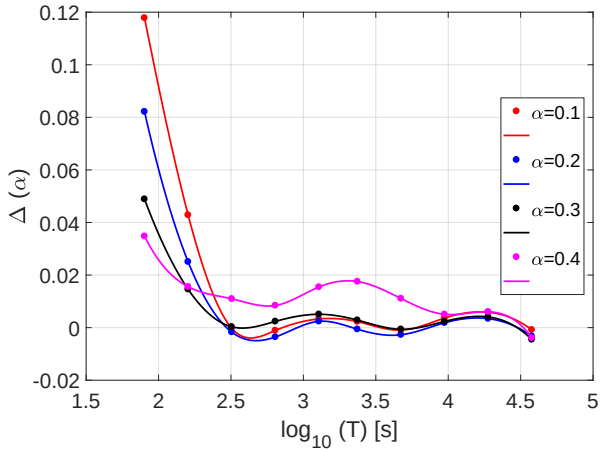


Fig. 6. Cost difference $\Delta(\alpha)$ versus accumulation interval $T$.

## VI. Conclusion

In this paper, we focused on the improvement bound of DC proactive management using predictive models. The main contribution of this paper is threefold: 1) the mathematical expression of the relative cost improvement brought by proactive management, 2) a generic methodology that can be applied to any DC, but the values of the obtained bounds depend on the data set considered, and 3) the improvement of outgoing jobs prediction using an ARMAX predictor. A data set provided by Google has been used to illustrate the procedure.

The used prediction model takes into account not only the time series to be predicted, but also the possibility of improving the performance using other correlated signals. From the Google data set studied, we conclude that:

- The larger the accumulation interval, the lower the cost, as the requests become more and more correlated are more predictable. On the other hand, the lower the accumulation interval, the less correlated the requests and the poorer the prediction.
- The maximum improvement is approximately 12% and is reached for small accumulation intervals, $T < 2.5$ mn. Then, the improvement is getting smaller when the accumulation interval increases, e.g. lesser than 2% for $T \geq 2.5$ mn.
- If some periodicity exists in job arrivals (e.g. 5 mn and 1 hour in the Google dataset), the Last Value predictor and the ARMA predictor give close results for such values of the accumulation interval $T$.
- We obtain a dynamic model of the data center to improve the predictions of the output flow of jobs using the ARMAX model. The improvements are explained in the context of Granger-causal signals.

In this paper, we showed it is possible to predict with a high-confidence degree the arrival of jobs and the resources they request (CPU and memory capacity, etc.). This makes it possible to have an almost optimal placement and scheduling of the jobs, leading to a near-best management of DCs that can allow significant energy savings at the global scale.

## REFERENCES


[1] M. Avgerinou, P. Bertoldi and L. Castellazzi, "Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency". Energies Journal, MDPI 2017.

[2] C. Reiss, J. Wilkes, J. L. Hellerstein, "Google cluster-usage traces: format + schema", Google Inc. Technical Report, Mountain View, CA, Nov 2011. https://github.com/google/cluster-data

[3] J. Wilkes, "More Google cluster data", Google research blog, Nov. 2011. http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html

[4] B. Liu, Yinan Lin and Y. Chen, "Quantitative workload analysis and prediction using Google cluster traces," 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, 2016, pp. 935-940.

[5] J. J. Prevost, K. Nagothu, B. Kelley and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," 2011 6th International Conference on System of Systems Engineering, Albuquerque, NM, 2011, pp. 276-281.

[6] Yoon MS, Kamal AE, Zhu Z (2017) Adaptive data center activation with user request prediction. Comput Netw 122:191-204.

[7] S. Mazumdar and A.S. Kumar, Forecasting Data Center Resource Usage: An Experimental Comparison with Time-Series Methods. Springer International Publishing AG 2018 A. Abraham et al. (eds.), Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016), Advances in Intelligent Systems and Computing 614, DOI 10.1007/978-3-319-60618-7 16

[8] N. Ben Hassine, R. Milocco, P. Minet, "ARMA based Popularity Prediction for Caching in Content Delivery Networks", Wireless Days 2017, IEEE Communications Society, Porto, Portugal, March 2017.

[9] M. Alam, K. A. Shakil, S. Sethi, "Analysis and Clustering of Workload in Google Cluster Trace based on Resource Usage", Jan. 2015,

[10] O. Beaumont, L. Eyraud-Dubois, J.A. Lorenzo-Del-Castillo, "Analyzing real cluster data for formulating allocation algorithms in Cloud platforms", 2nd International Symposium on Computer architecture and High Performance Computing (SBAC-PAD), Paris, France, Oct. 2014.

[11] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis", ACM Symposium on Cloud Computing (SoCC), San Jose, CA, Oct. 2012,

[12] S. Yousif, A Al-Dulaimy, "Clustering Cloud Workload Traces to Improve the Performance of Cloud Data Centers", Proceedings of the World Congress on Engineering 2017, (WCE2017), July 2017.

[13] S. Di, D. Kondo, W. Cirne, "Host Load Prediction in a Google Compute Cloud with a Bayesian Model", Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC'12, Salt Lake City, Utah, IEEE Computer Society Press, 2012.

[14] P. Minet, E. Renault, I. Khoufi, S. Boumerdassi, "Analyzing traces from a Google data center", 14th International Wireless Communications and Mobile Computing Conference (IWCMC 2018), Limassol, Cyprus, June 2018.

[15] C. Liu, Y. Wan, L. Tian, Y. Zhou, J. Shi, "Base station sleeping control with energy-stability tradeoff in centralized radio access networks", 2015 IEEE Global Communications Conference (GLOBECOM), Dec 2015.

[16] K. Boulos, M. El Helou, M. Ibrahim, K. Khawam, H. Sawaya, S. Martin,"Interference-aware clustering in cloud radio access networks", 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), Prague, Czech Republic, Sept. 2017.

[17] Dimitris G. Manolakis, Vinay K. Ingle, Stephen M. Kogon : Statistical and Adaptive Signal Processing, McGraw-Hill,

[18] L. Ljung, "System Identification: Theory for the User", Upper Saddle River, NJ, Prentice-Hal PTR, 1999. See chapter about computing the estimate.

[19] C. W. J. Granger, "Investigating Causal Relations by Econometric Models and Cross-spectral Methods", Econometrica 37 (3): 424-438, 1969.